**Binary Search**

Binary Search (Intro)

Properties: -

1. Binary Search can be applied only on the sorted array.
2. Found the middle element of the array i.e. Mid = (start + end)/2
3. If the Key element is greater than the mid element then Search right side.
4. Otherwise (If Key elements is less than the mid element) then Key element is found left side, we must search the left side of the mid.

Start 0

end

0 1 2 3 4

2 4 6 8 10

Here, start = 0, end = n-1 i.e. 4 Key = 4

mid = (start + end)/2

Solution.

mid = (0+4)/2 = 2 (not found)

here key is less than array of mid so,

mid = (0+1)/2 = 1 (Found) at index 1

**Question 1**. Write a CPP Program to find the element Using the Binary Search Algorithm.

#include<iostream>

using namespace std;

void binarySearch(int arr[], int n, int target){

  int start = 0, end = n-1;

  while(start <= end){

    int mid = start + (end - start)/2;

    if(arr[mid] == target){

      cout << "Target element " << target << " Got it! at index " << mid;

      break;

    } else if(arr[mid] < target){

      start = mid+1;

    } else {

      end = mid-1;

    }

  }

}

int main() {

  int arr[1000];

  int size, k;

  cout << "Enter the size of the array: ";

  cin >> size;

  cout << "Input the elements into the array: ";

  for (int i = 0; i < size; i++){

    cin >> arr[i];

  }

  cout << "Enter the searching element: ";

  cin >> k;

  // Binary Search

  binarySearch(arr, size, k);

}

/\*

Output:-

--------

Enter the size of the array: 4

Input the elements into the array: 10 20 40 50

Enter the searching element: 40

Target element 40 Got it! at index 2

Enter the size of the array: 5

Input the elements into the array: 1 20 30 40 55

Enter the searching element: 55

Target element 55 Got it! at index 4

\*/

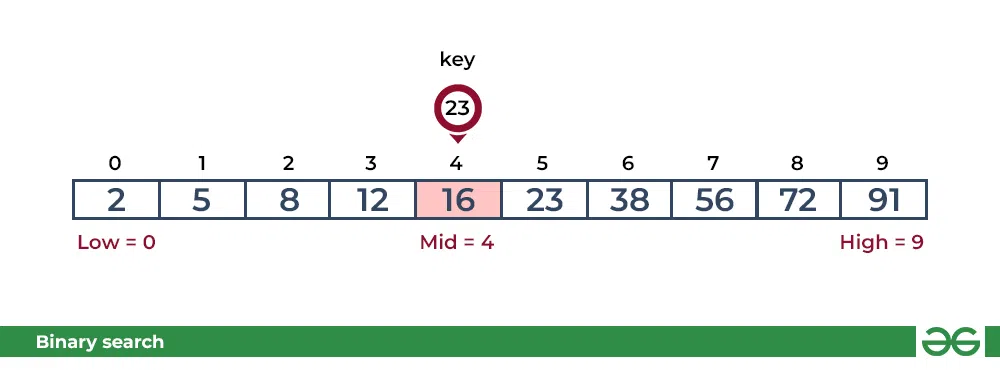
## How does Binary Search work?

To understand the working of binary search, consider the following illustration:

*Consider an array****arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91}****, and the****target = 23****.*

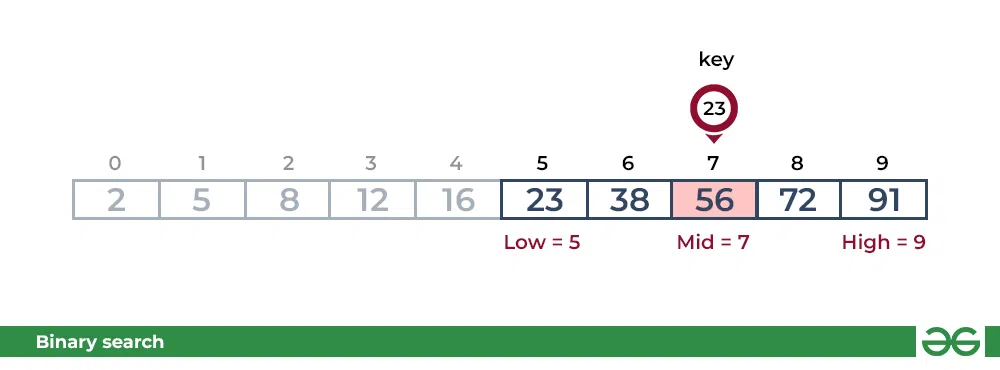
***First Step:****Calculate the mid and compare the mid element with the key. If the key is less than mid element, move to left and if it is greater than the mid then move search space to the right.*

* *Key (i.e., 23) is greater than current mid element (i.e., 16). The search space moves to the right.*



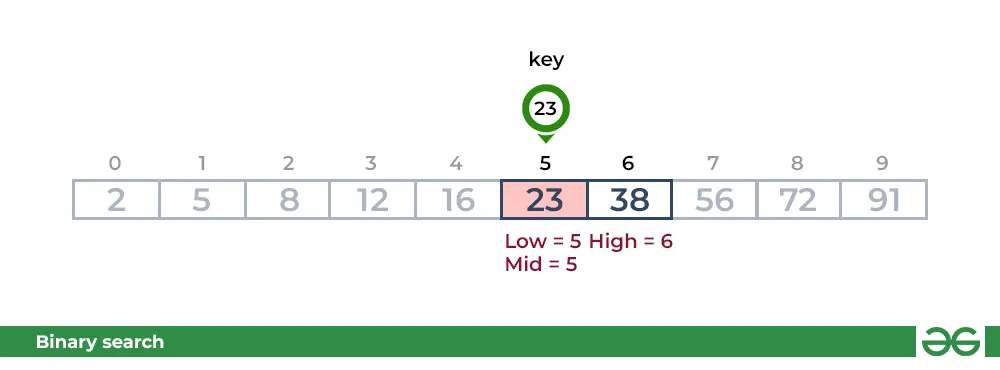
Binary search algorithm: Compare key with 16

* *Key is less than the current mid 56. The search space moves to the left.*

**

Binary search algorithm: Compare key with 56

***Second Step***: If the key matches the value of the mid element, the element is found and stop search.



Binary search: Key matches with mid

**Question:**

Given a sorted and rotated array A of N distinct elements which is rotated at some point, and given an element key. The task is to find the index of the given element key in the array A. The whole array A is given as the range to search.

Example 1:

**Input:**

N = 9

A[] = {5, 6, 7, 8, 9, 10, 1, 2, 3}

key = 10

l = 0 , h = 8

**Output**:

5

**Explanation**: 10 is found at index 5.

**Example 2**:

**Input**:

N = 4

A[] = {3, 5, 1, 2}

key = 6

l = 0 , h = 3

**Output**:

-1

**Explanation**: There is no element that has value 6.

Your Task:  
Complete the function search() which takes an array arr[] and start, end index of the array and the K as input parameters, and returns the answer.  
  
Can you solve it in expected time complexity?

Expected Time Complexity: O(log N).  
Expected Auxiliary Space: O(1).

Constraints:  
1 ≤ N ≤ 107  
0 ≤ A[i] ≤ 108  
1 ≤ key ≤ 108

Solution.

#include<iostream>

using namespace std;

int binarySearch(int arr[], int n, int target){

  int start = 0, end = n-1;

  while(start <= end){

    // int mid = (start + end )/2;

    int mid = start + (end - start)/2;

    if(arr[mid] == target){

      cout << "Target element " << target << " Got it! at index " << mid;

      return mid;

    } else if(arr[mid] < target){

      start = mid+1;

    } else {

      end = mid-1;

    }

  }

  // If we reach here, the element was not present then

  return -1;

}

int main() {

  int arr[1000];

  int size, k;

  cout << "Enter the size of the array: ";

  cin >> size;

  cout << "Input the elements into the array: ";

  for (int i = 0; i < size; i++){

    cin >> arr[i];

  }

  cout << "Enter the searching element: ";

  cin >> k;

  // Binary Search

  int result = binarySearch(arr, size, k);

  (result == -1) ? cout << "Element is not present in array"

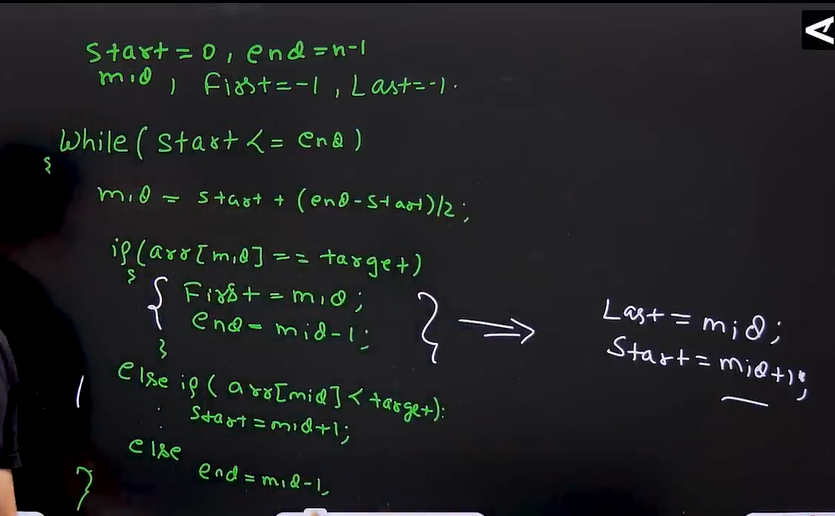
          : cout << " (Element is present in the array) ";

  return 0;

}

Note: - To avoid the overflow condition like integer overflow, we can find the mid using the following approach.

mid = start + (end-start)/2 | = end + (start-end)/2

Question FL. First and Last Occurrence of Integer sorted array using the binary search approach.  
  


First & Last occurrence of integer repeated array